A general approach to pluggable workflow-processing for the next generation UNICORE

Henning Mersch*, Bernd Schuller, Achim Streit

Forschungszentrum Jülich
in der Helmholtz-Gemeinschaft

Abstract

Workflow processing in distributed systems and Grids includes a huge range of activities today. Running complex applications in a service-oriented environment requires both low-level service orchestration and a domain-oriented way to define the high-level application workflow.
As shown by the multitude of existing workflow languages and approaches, it is next to impossible to choose a single approach to workflow processing that will fit all the needs.
We expect many workflow representations to coexist in a typical ``next-generation'' Grid. Here we present parts of a flexible, generic system for workflow processing that is fairly independent of the possible description languages and workflow processing engines that are used.
This work is motivated by the need for a versatile end-user client for the next generation of the UNICORE Grid system [1], which might be available as domain specific and / or a grid expert version.

Introduction

The UNICORE 6 Grid system [2] defines a set of low-level services, such as job execution and file transfer.
Next generation Grid solutions will have to provide user-friendly interfaces for getting accepted in a wider range of customers.
Existing client software for UNICORE 6 is focused on providing low-level access to grid-enabled services, suitable for the ``Grid expert'' user. However, in a lot of relevant use cases, the users are non-Grid experts, who will be daunted by such a client. Thus we suggest to provide domain specific client applications hiding the Grid-related complexity.
Effective development of these clients requires a well defined infrastructure giving as much high level functionality to the client application developer as possible
Intel's GridBeans [3] approach offers a framework for developing and reusing code, by encapsulating the Grid enabled application by providing user interfaces based upon different technologies such as Swing. Thus, GridBeans will provide a well defined and already widely used basis for our approach.

Workflow Representation and Domain-Specific-Language

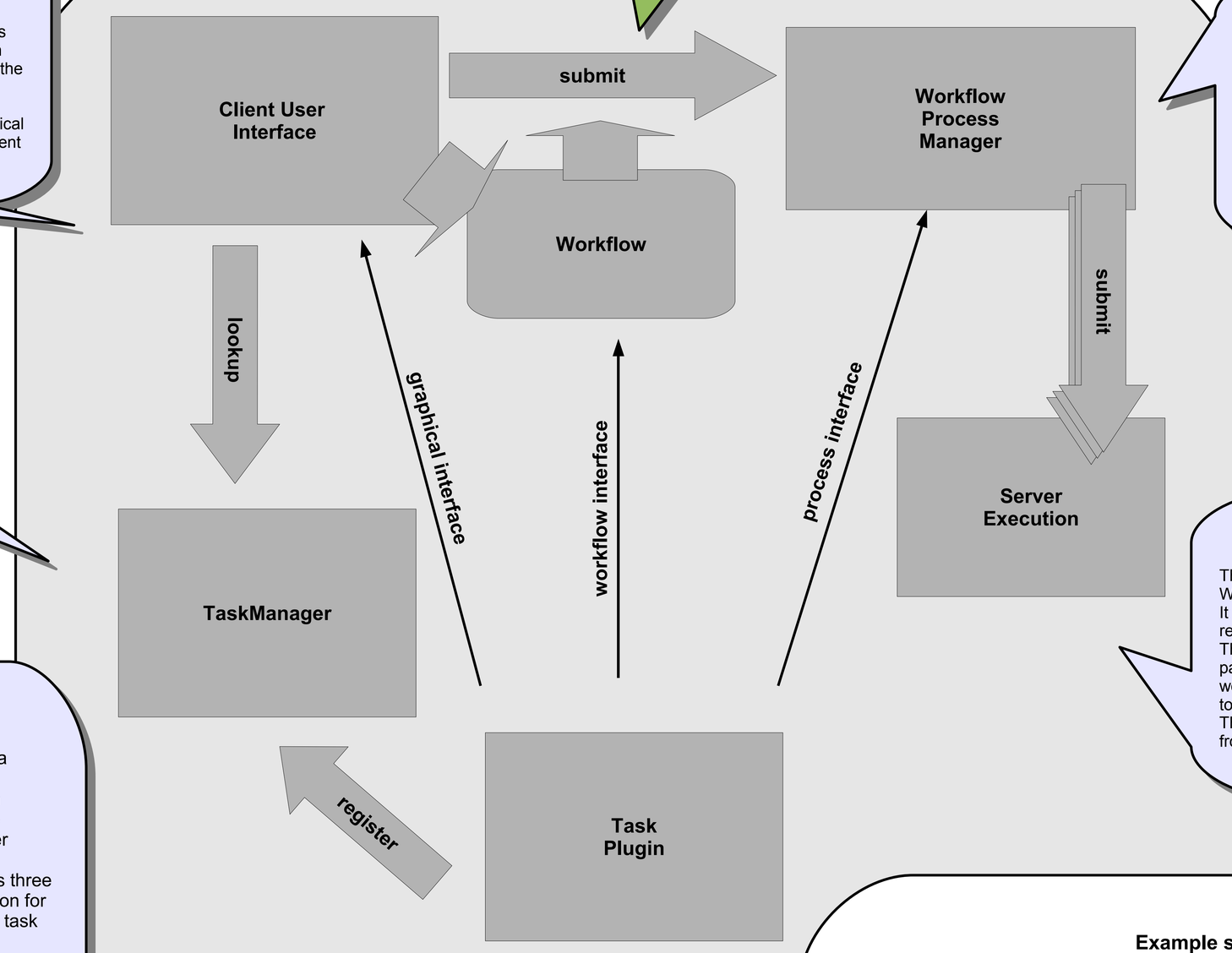
Many workflow description languages already exist, such as BPEL [4], which was originally intended for web service orchestration. In our approach, BPEL could be used to build a generic workflow processing environment.
Nevertheless it is often beneficial to use a **domain specific language**, which can be tailored to a specific application domain. This might in turn rely on existing workflow description representations by adding own task descriptions. This additional abstraction layer would reduce the amount on both client and server side for implementation. Also the processing of a domain specific workflow might be solve special purposes of the domain

Chemomentum

Chemomentum [5] will provide an integrated Grid solution for workflow-centric, complex applications with a focus on data management and knowledge. It will place the end users into the focus, enabling them to use powerful tools in a natural and transparent fashion.
The project will provide Grid-enabled applications, data services and knowledge management solutions, offering integrated decision support services for risk assessment, toxicity prediction and drug design.

User Client Interface

The user (as synonym for a client application) provides a workflow to the system. The workflow is submitted to the WorkflowProcessManager, which keeps track of the execution process and informs the user about state changes - thus these two must share an interface definition.
The WorkflowProcessManager updates the graphical representation of the workflow by informing the client about upcoming events.



WorkflowProcessManager

The WorkflowProcessManager builds the workflow by collecting the contributions from the individual TaskPlugins and dependencies. It interprets or converts this to a supported language of the ServerStub. Several scenarios for workflow processing are conceivable.
The actual workflow processing may rely on an external workflow processing engine, such as for example jBPM [6] or ActiveBPEL [7].

TaskManager

This component manages all known task types. Thus the TaskManager can provide the workflow to the WorkflowProcessManager, which is able to execute it.

Server communication

The ServerStub component depends on the WorkflowProcessManager or at least upon its type.
It is essential for the system that the chosen workflow representation is easily replacable.
The WorkflowProcessManager could (and in most cases should) be part of the server side because it often is not feasible to process a workflow from the client. At least a management for communication to different ServerStubs should be implemented on client side.
Thus, the ServerStub could be minimal - just serving a connection from the WorkflowProcessManager to receive the tasks to process.

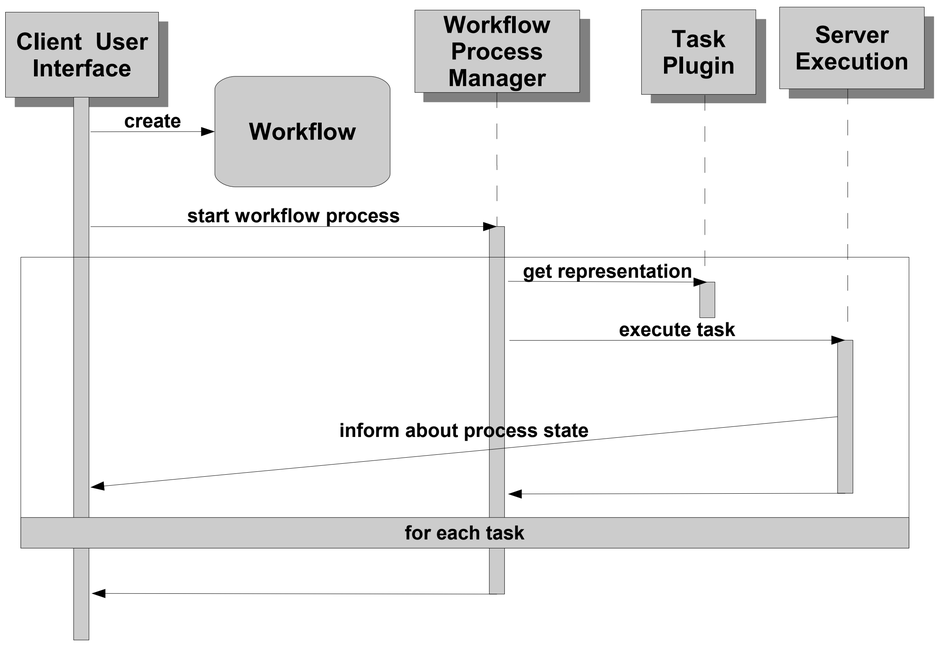
Pluggable Tasks

Different task types (for example executing a script) are represented by TaskPlugins.
These are components implementing plugin interfaces and registering themselves to the TaskManager, which is part of the client user interface component.
For a given task type, a TaskPlugin provides three components: firstly, a graphical representation for integration into the user interface. Second a task description for processing the task for the WorkflowProcessManager and third a representation of this task as part of a workflow.

Pluggable Dependencies

A workflow can be extended by complex dependencies as well. Thus, even if not mentioned directly here graphically, dependencies are made extensible, too.
For doing so, a extension has to be representable in terms of the workflow representation used by the WorkflowProcessManager. Domain specific dependencies could be developed and added later on the same way like TaskPlugins are.

Example sequence diagram



Chemomentum

To create, edit and run complex, multi-step user-level applications like in Chemomentum in a simple and intuitive fashion, the end user interfaces (workstation client software as well as web portals) will hide the Grid related complexity and let the users deal with application specific concepts and artefacts they know about.
User applications providing this high level abstraction will produce domain specific workflows, which are defined in a domain specific language.
These workflows could be processed via the presented system.

Conclusion

The definition of the interfaces is the most important part of an implementation. Even if the system initially uses certain fixed formats and representations, new standards which might have to be taken into account arise rapidly in todays fast growing Grid technologies. So, well defined interfaces are a major factor for future extensions and easy adaptability.This system is highly flexible due to the architecture and not constrained by the suggested standards.

We presented a system, which targets the process of workflow execution in an distributed and heterogeneous Grid environment. Building up an extensible workflow client application enables us to develop a rich next generation UNICORE client, which targets Grid-expert, as well as domain specific solutions for domain experts. This workflow representation, which will be processed by the WorkflowProcessManager using an external processing engine.
The separated tasks are send for execution to the ServerStub, which keeps the client up-to-date about the state.Tasks and control constructs can be plugged in using the above mentioned plugin mechanism or the existing GridBean technology making this system adaptable to any needs of a experienced Grid-expert.

References

[1] A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and Ph. Wieder. **UNICORE - From Project Results to Production Grids.** L. Grandinetti (Ed.), *Grid Computing: The New Frontiers of High Performance Processing*, Advances in Parallel Computing, Vol. 14, Elsevier, 2005, pages 357-376
[2] Unicore 6: 12 October 2006 <http://www.unicore.eu>
[3] R. Ratering, M. Riedel, A. Vanni, K. Benedyczak, A. Lukichev, D. Mallmann, G. Ohme, C. Cacciari, P. Bala, S. Lanzarini, M. Borcz, R. Kluszcynski. **GridBeans: Supporting e-Science and Grid Applications**, *e-Science 2006, to appear*
[4] S. Thatte (Ed.), **Business Process Execution Language for Web Services version 1.1**, 12 October 2006 <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>
[5] Chemomentum, 12 October 2006 <http://www.chemomentum.org>
[6] JBoss jBPM, 12 October 2006 <http://www.jboss.com/products/jbpm>
[7] ActiveBPEL, 12 October 2006 <http://www.activebpel.org>

* Email:h.mersch@fz-juelich.de